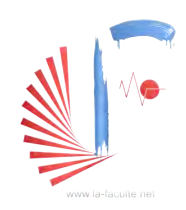
In the name of Allah, the Most Gracious, the Most Merciful






Algorithmique

Notions et instructions de base

Tests: instructions conditionnelles (1)

- Les instructions conditionnelles servent à n'exécuter une instruction ou une séquence d'instructions que si une condition est vérifiée
- On utilisera la forme suivante: **Si** condition **alors**

  instruction ou suite d'instructions1

  Sinon

  instruction ou suite d'instructions2

  Finsi

  - la condition ne peut être que vraie ou fausse
  - si la condition est vraie, se sont les instructions1 qui seront exécutées
  - si la condition est fausse, se sont les instructions2 qui seront exécutées
  - la condition peut être une condition simple ou une condition composée de plusieurs conditions

Tests: instructions conditionnelles (2)

- La partie Sinon n'est pas obligatoire, quand elle n'existe pas et que la condition est fausse, aucun traitement n'est réalisé
  - On utilisera dans ce cas la forme simplifiée suivante:

Si condition **alors**

instruction ou suite d'instructions1

Finsi

Exemple (Si...Alors...Sinon)

```
Algorithme AffichageValeurAbsolue (version1)
Variable x : réel
Début
    Ecrire (" Entrez un réel : “)
    Lire (x)
    Si (x < 0) alors
        Ecrire ("la valeur absolue de ", x, "est:",-x)
    Sinon
        Ecrire ("la valeur absolue de ", x, "est:",x)
    Finsi
Fin
```

Exemple (Si…Alors)

```
Algorithme AffichageValeurAbsolue (version2)
Variable x,y : réel
Début
        Ecrire (" Entrez un réel : “)
        Lire (x)
         y← x
        Si (x < 0) alors
                y ← -1*x
        Finsi
        Ecrire ("la valeur absolue de ", x, "est:",y)
Fin
```

Conditions composées

- Une condition composée est une condition formée de plusieurs conditions simples reliées par des opérateurs logiques:

  ET, OU, NON

- Exemple :
  - x compris entre 2 et 6 : (x > 2) ET (x < 6)

- L'évaluation d'une condition composée se fait selon des règles présentées généralement dans ce qu'on appelle tables de vérité

Tables de vérité

C1	C2	**C1 ET C2**
VRAI	VRAI	**VRAI**
VRAI	FAUX	**FAUX**
FAUX	VRAI	**FAUX**
FAUX	FAUX	**FAUX**

C1	C2	**C1 OU C2**
VRAI	VRAI	**VRAI**
VRAI	FAUX	**VRAI**
FAUX	VRAI	**VRAI**
FAUX	FAUX	**FAUX**

C1	**NON C1**
VRAI	**FAUX**
FAUX	**VRAI**

Tests imbriqués

- Les tests peuvent avoir un degré quelconque d'imbrications

```
Si condition1 alors
        Si condition2 alors
                instructionsA
        Sinon
                instructionsB
        Finsi
Sinon
        Si condition3 alors
                instructionsC
        Finsi
Finsi
```

Tests imbriqués: exemple (version 1)

```
Variable n : entier
Début
    Ecrire ("entrez un nombre : ")
    Lire (n)
    Si (n < 0) alors
        Ecrire ("Ce nombre est négatif")
    Sinon
        Si (n = 0) alors
            Ecrire ("Ce nombre est nul")
        Sinon
            Ecrire ("Ce nombre est positif")
        Finsi
    Finsi
Fin
```

Tests imbriqués: exemple (version 2)

```
Variable n : entier
 Début
        Ecrire ("entrez un nombre : ")
        Lire (n)
        Si (n < 0) alors    Ecrire ("Ce nombre est négatif")
        Finsi
        Si (n = 0) alors    Ecrire ("Ce nombre est nul")
        Finsi
        Si (n > 0) alors    Ecrire ("Ce nombre est positif")
        Finsi
Fin
```

Remarque : dans la version 2 on fait trois tests systématiquement alors que dans la version 1, si le nombre est négatif on ne fait qu'un seul test

Conseil : utiliser les tests imbriqués pour limiter le nombre de tests et placer d'abord les conditions les plus probables

Tests imbriqués: exercice

Le prix de photocopies dans une reprographie varie selon le nombre demandé: 0,5 DH la copie pour un nombre de copies inférieur à 10, 0,4DH pour un nombre compris entre 10 et 20 et 0,3DH au-delà.

Ecrivez un algorithme qui demande à l'utilisateur le nombre de photocopies effectuées, qui calcule et affiche le prix à payer

Tests imbriqués: corrigé de l'exercice

```
Variables copies : entier
          prix : réel
Début
     Ecrire ("Nombre de photocopies : ")
     Lire (copies)
     Si (copies < 10) Alors
        prix ← copies*0.5
     Sinon Si (copies) < 20
               prix ← copies*0.4
           Sinon
               prix ← copies*0.3
           Finsi
     Finsi
     Ecrire (“Le prix à payer est : ”, prix)
Fin
```

Instructions itératives: les boucles

- Les boucles servent à répéter l'exécution d'un groupe d'instructions un certain nombre de fois
- On distingue trois sortes de boucles en langages de programmation :
  - Les **boucles tant que** : on y répète des instructions tant qu'une certaine condition est réalisée
  - Les **boucles jusqu'à** : on y répète des instructions jusqu'à ce qu'une certaine condition soit réalisée
  - Les **boucles pour** ou avec compteur : on y répète des instructions en faisant évoluer un compteur (variable particulière) entre une valeur initiale et une valeur finale

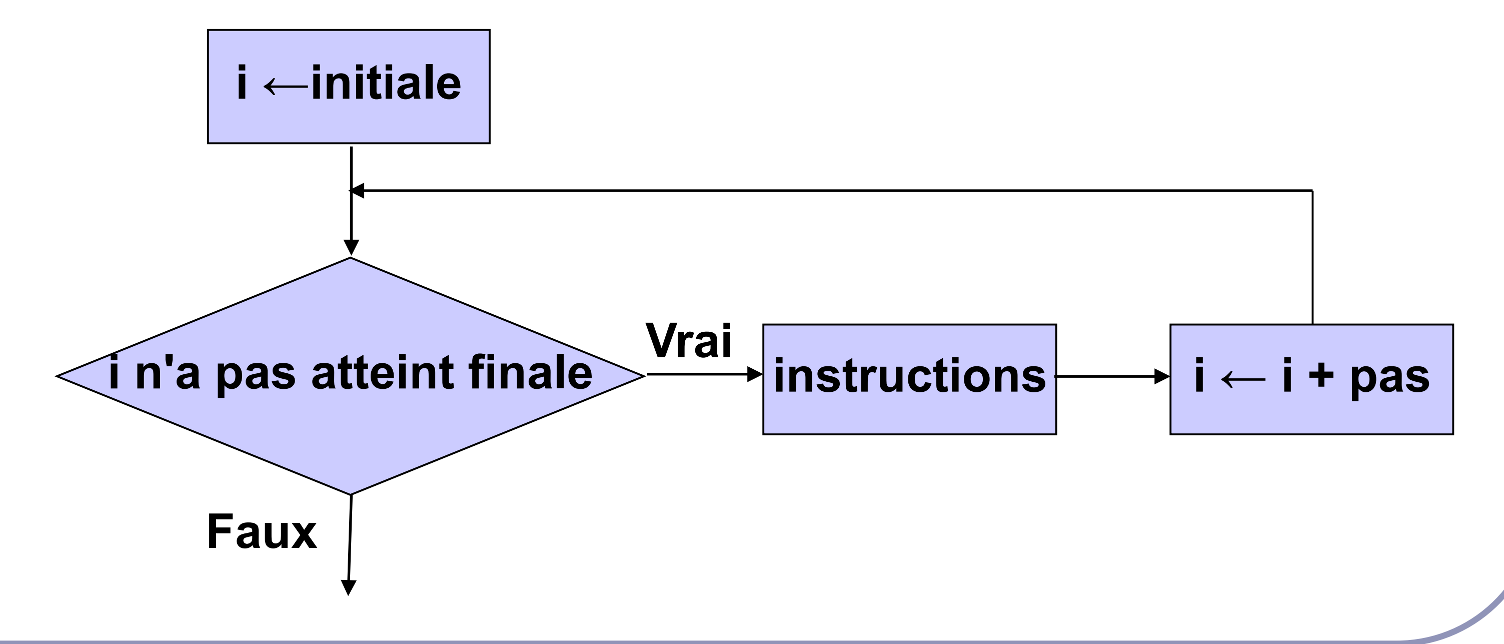
(Dans ce cours, on va s'intéresser qu'à la boucle *Pour*)

Les boucles Pour

Pour compteur **allant de** initiale **à** finale par **pas** valeur du pas

instructions

FinPour



Les boucles Pour

- Remarque : le nombre d'itérations dans une boucle Pour est connu avant le début de la boucle

- **Compteur** est une variable de type entier (ou caractère). Elle doit être déclarée

- **Pas** est un entier qui peut être positif ou négatif. **Pas** peut ne pas être mentionné, car par défaut sa valeur est égal à 1. Dans ce cas, le nombre d'itérations est égal à finale - initiale+ 1

- **Initiale et finale** peuvent être des valeurs, des variables définies avant le début de la boucle ou des expressions de même type que compteur

Déroulement des boucles Pour

1) La valeur initiale est affectée à la variable compteur

2) On compare la valeur du compteur et la valeur de finale :

   a) Si la valeur du compteur est > à la valeur finale dans le cas d'un pas positif (ou si compteur est < à finale pour un pas négatif), on sort de la boucle et on continue avec l'instruction qui suit FinPour

   b) Si compteur est <= à finale dans le cas d'un pas positif (ou si compteur est >= à finale pour un pas négatif), instructions seront exécutées

      i. Ensuite, la valeur de compteur est incrémentée de la valeur du pas si pas est positif (ou décrémenté si pas est négatif)

      ii. On recommence l'étape 2 : La comparaison entre compteur et finale est de nouveau effectuée, et ainsi de suite …

Boucle Pour : exemple1

Ecrire un algorithme qui calcule de x à la puissance n où x est un réel non nul et n un entier positif ou nul

Boucle Pour : exemple1

Calcul de x à la puissance n où x est un réel non nul et n un entier positif ou nul

```
Variables x, puiss : réel
          n, i : entier
Debut
     Ecrire (" Entrez la valeur de x ")
     Lire (x)
     Ecrire (" Entrez la valeur de n ")
     Lire (n)

     puiss ← 1
     Pour i allant de 1 à n
               puiss← puiss*x
     FinPour
     Ecrire (x, " à la puissance  ", n, "  est égal à  ", puiss)
Fin
```

Boucle Pour : exemple1 (version 2)

Calcul de x à la puissance n où x est un réel non nul et n un entier positif ou nul (**version 2 avec un pas négatif**)

```
Variables x, puiss : réel
          n, i : entier
Debut
    Ecrire (" Entrez respectivement les valeurs de x et n ")
    Lire (x, n)
    puiss ← 1
    Pour i allant de n à 1 par pas -1
        puiss← puiss*x
    FinPour
    Ecrire (x, " à la puissance  ", n, "  est égal à  ", puiss)
Fin
```

Boucle Pour : remarque

- Il faut éviter de modifier la valeur du compteur (et de finale) à l'intérieur de la boucle. En effet, une telle action :
  - perturbe le nombre d'itérations prévu par la boucle Pour
  - rend difficile la lecture de l'algorithme
  - présente le risque d'aboutir à une boucle infinie

Exemple : **Pour** i **allant de** 1 **à** 5

i ← i -1

écrire(" i = ", i)

Finpour